

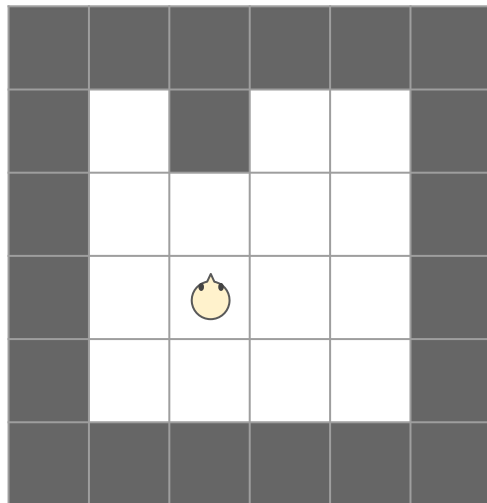
関数説明資料

方角の考え方(要修正)

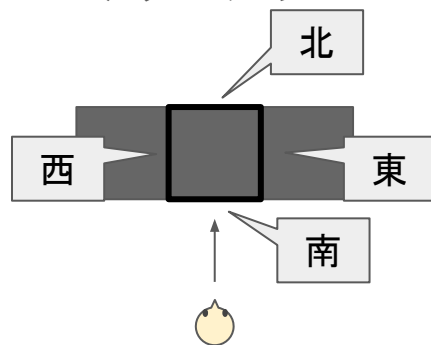
マップ

1	1	1	1	1	1
1	0	1	0	0	1
1	0	0	0	0	1
1	0	N	0	0	1
1	0	0	0	0	1
1	1	1	1	1	1

プレイヤー

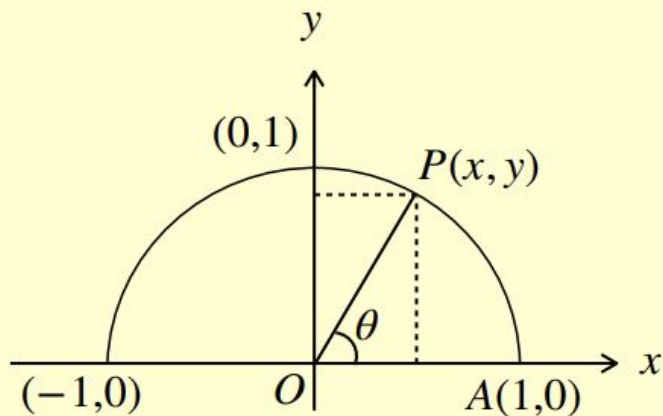


テクスチャー



※プレイヤーが北を向いたときに南のテクスチャになる

三角比の定義



このとき、 θ の三角比を次のように定義する。

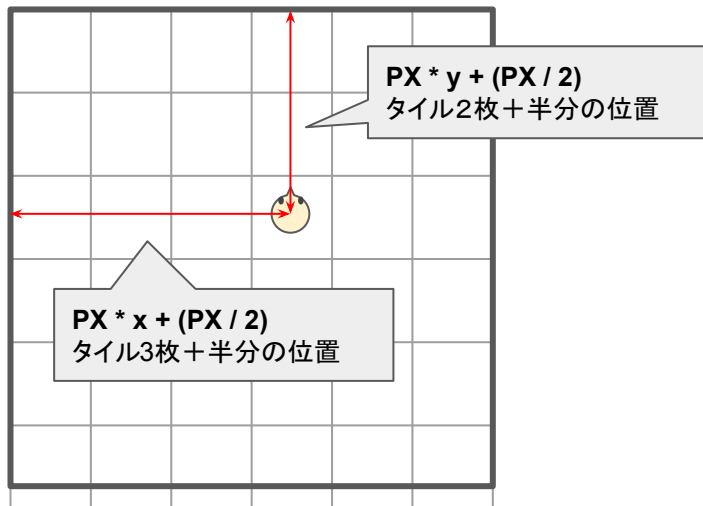
$$\sin \theta = y \quad \cos \theta = x \quad \tan \theta = \frac{y}{x}$$

(直線の傾き)

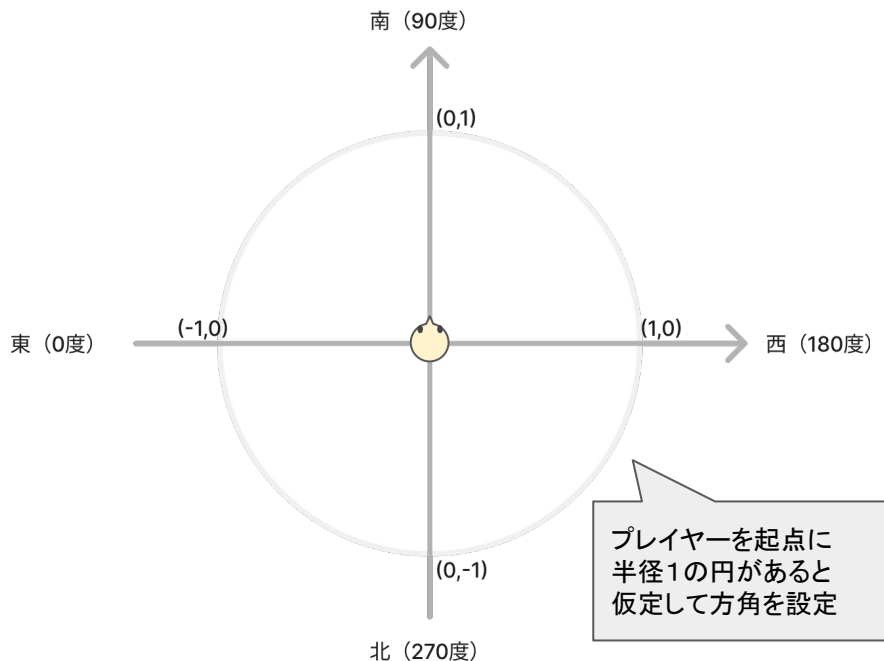
set_player_position関数

目的1: 画像上の座標を算出

例) map[2][3]の位置の場合



目的2: プレイヤーの方角を設定



set_player_angle

目的: プレイヤーの向きをラジアンで設定する

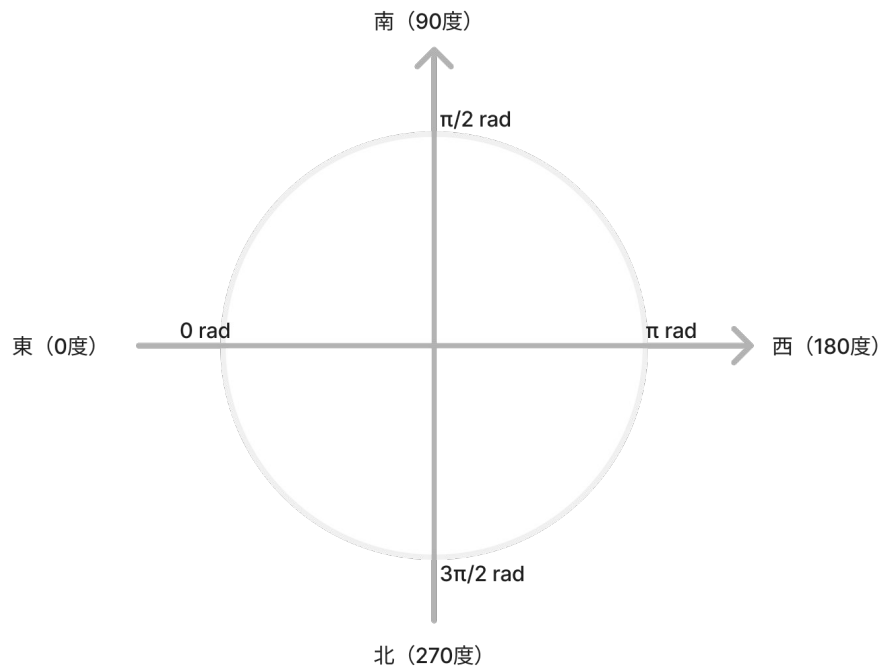
円の1周(360°) = 2π ラジアン

東(E): 0 ラジアン(0°)

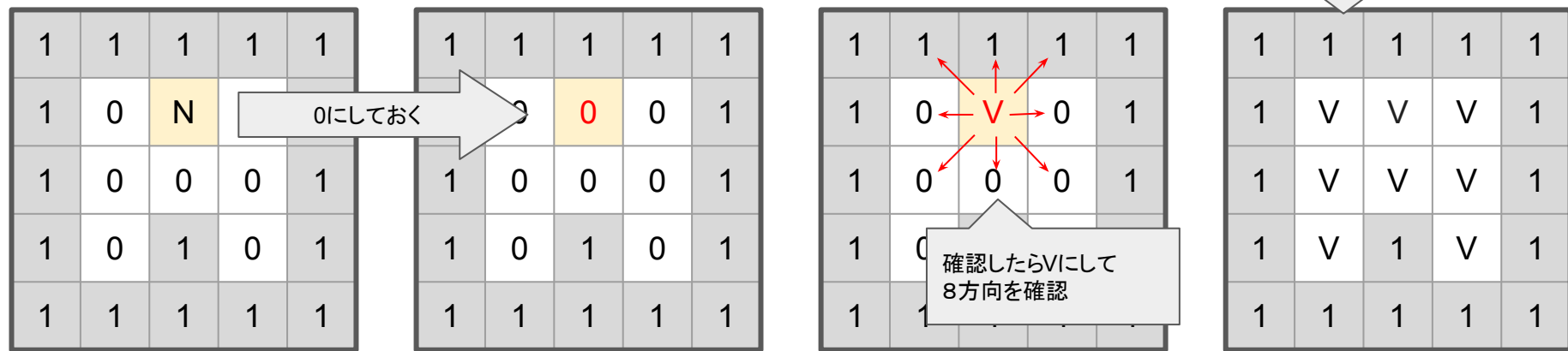
南(S): $\pi/2$ ラジアン(90°)

西(W): π ラジアン(180°)

北(N): $3\pi/2$ ラジアン(270°)



flood_fill



1. プレイヤーの位置を「0」に置き換える
2. プレイヤーの位置を起点に再帰的に探索を開始し、8方向を確認する
3. 確認したら「V」に書き換える
4. 確認箇所が1またはチェック済み(V)なら正常終了、
途中でスペースや不正な文字があったり、1で囲まれてなければエラーを返して探索終了

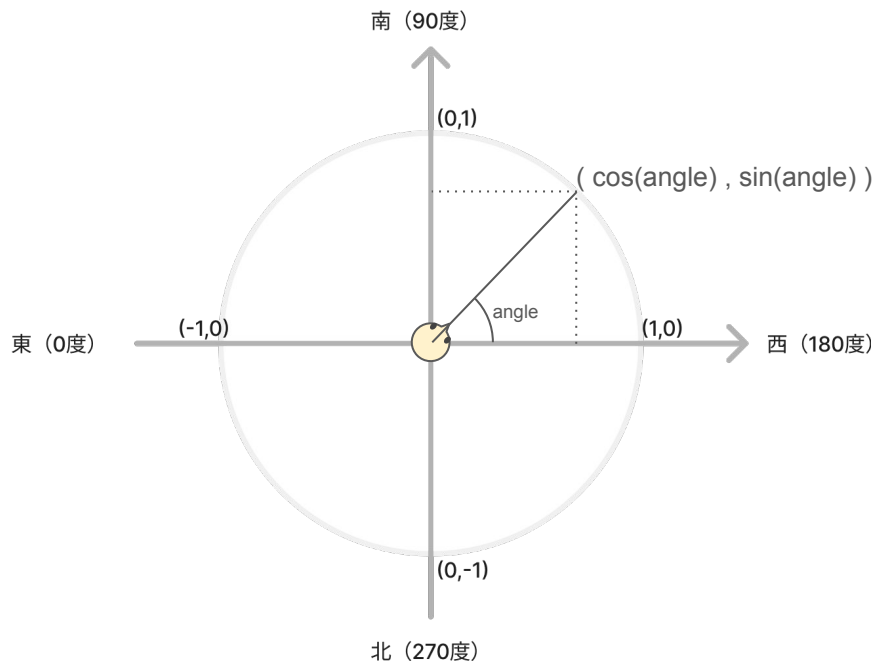
handle_left_angle / handle_right_angle

目的: プレイヤーの向きを更新する

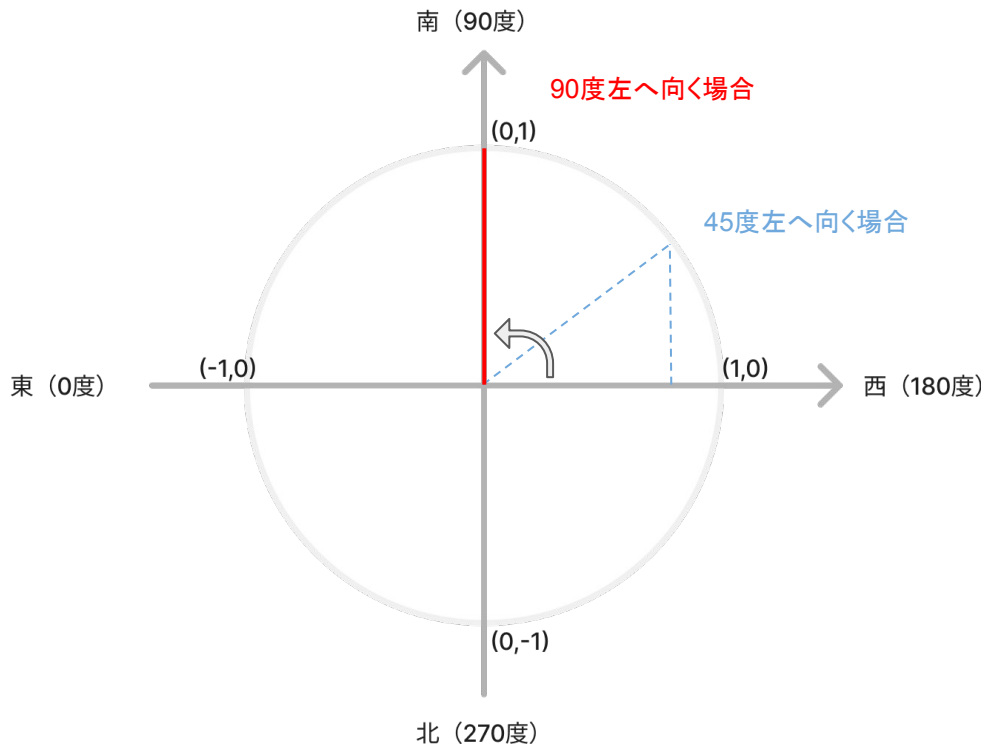
三角比の定義より、変更後の向きは

$$y = \sin(\text{angle})$$

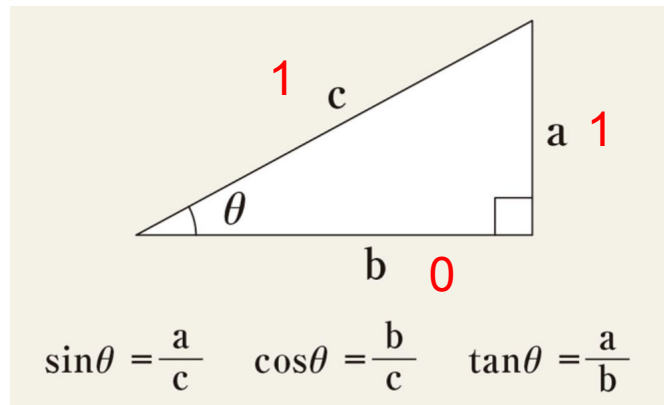
$$x = \cos(\text{angle})$$



補足：向きの計算

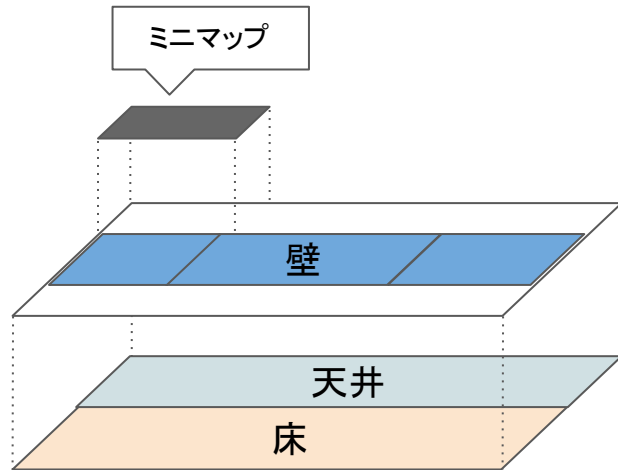


1. 西向き(180度)に立っている
2. 90度左へ向く
3. $90^\circ = \pi/2$ ラジアン $= 3.14/2 = 1.57$
4. $\cos(90^\circ) = 0$ 、 $\sin(90^\circ) = 1$
5. (0,1)なので南を向く



rendering

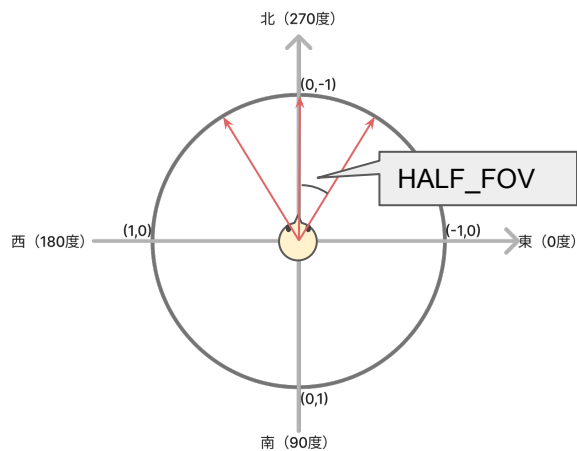
1. レイ毎の壁の高さを計算 [raycasting]
2. 床と天井を描画 [draw_floor_and_ceiling]
3. 壁を描画 [draw_walls]
4. ミニマップを描画(オプション)[draw_minimap]
5. ウィンドウにイメージを配置



raycasting

目的: プレイヤーの視界の左端から右端までレイを投射し、壁の位置を確認する

1. レイを投射し、壁までの距離を計算する [castray]
2. 魚眼補正を行う

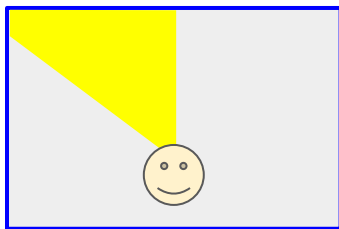
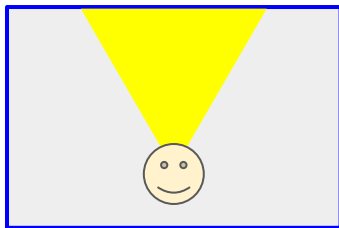


1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1

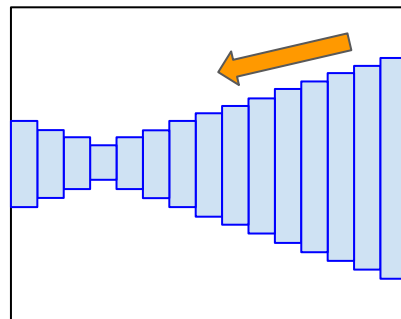
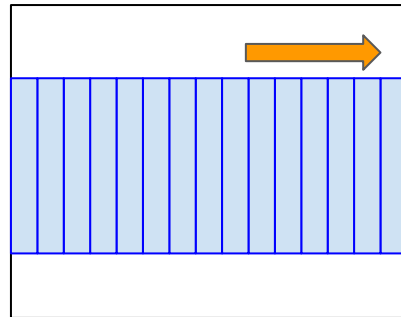
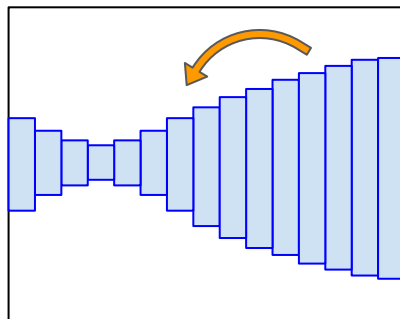
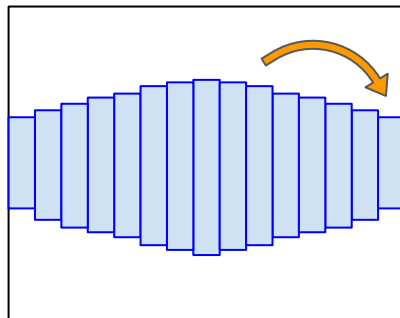
A 5x5 grid representing a map. The grid contains 1s (walls) and 0s (open space). A yellow circle is located at the intersection of the third row and fourth column. Four red arrows originate from this circle, pointing towards the top-right corner of the grid. A yellow shaded area is visible in the top-right corner, representing the field of view.

魚眼効果の補正

鳥瞰図

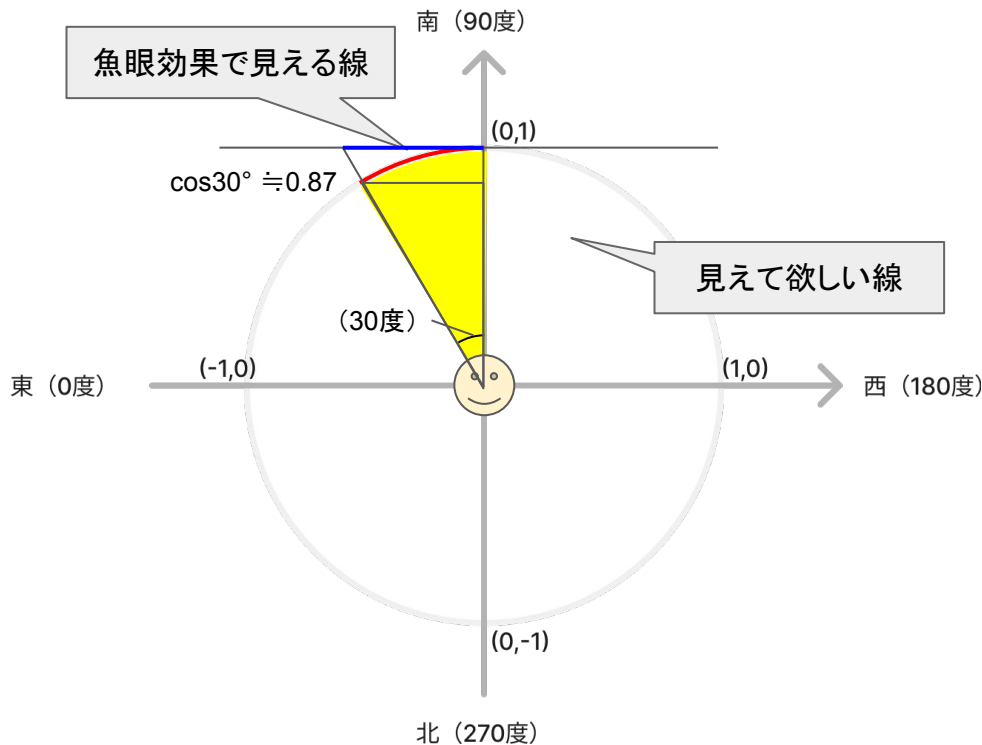


ゲームウィンドウ



プレイヤーからの距離を反映させた時、魚眼効果で遠くなるほど湾曲して見えてしまうため、これを真っ直ぐな線で表現したい

魚眼効果の補正計算



1. 南向き(90度)に立っている
2. 視界の左端が60度を向いている場合、
 $90\text{度} - 60\text{度} = 30\text{度}$
3. 何%距離を縮めればいいか？
 $\cos(0\text{度}) = 1$ ※正面の場合
 $\cos(30\text{度}) = 0.866$
4. 壁からの距離の約87%として考える

castray

目的: 壁までの距離を取得する

1. 次のグリッドまでのx・y方向の距離を取得

[next_grid_distance_y/x]

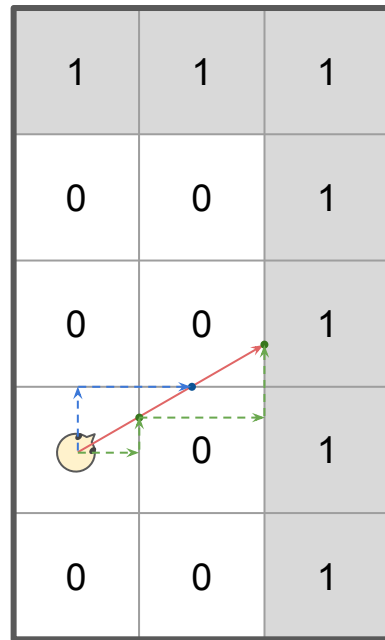
2. レイの位置を進める [increment_ray_length]

3. 壁のセルに当たったか確認 [check_wall]

4. 当たってなければ1～3を繰り返す

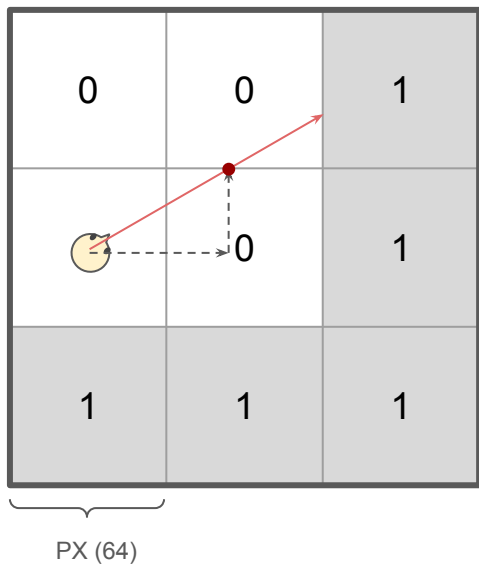
当たっていればどの方向の壁か取得し、位置を計算

[wall_collision] [wall_point]

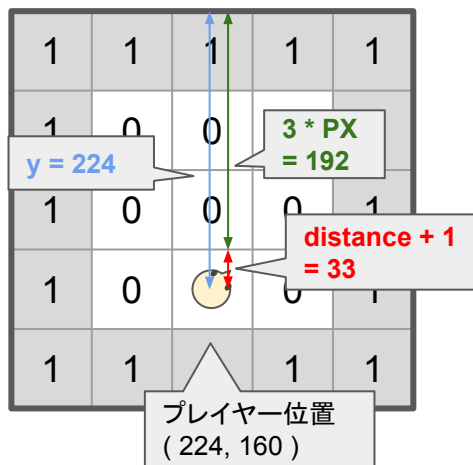


next_grid_distance_y(next_grid_distance_x)

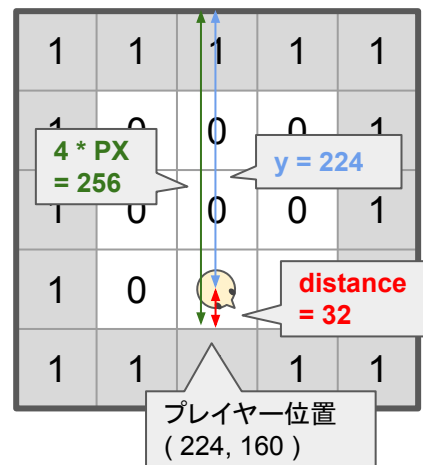
目的: プレイヤーの位置から次の水平線グリッドまでの距離を取得する



例)map[3][2]の位置、上(北)向きの場合



例)map[3][2]の位置、下(南)向きの場合



※1はグリッド線

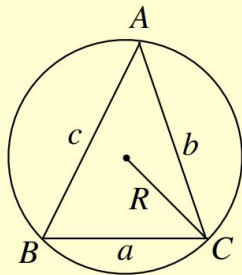
正弦定理



正弦定理

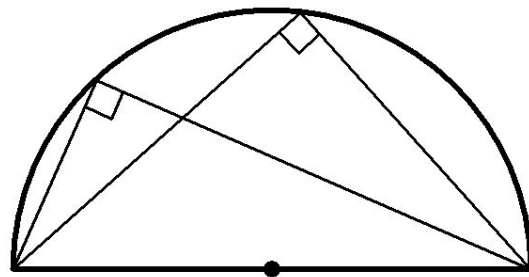
$\triangle ABC$ の外接円の半径 R とすると、次の関係が成り立つ

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = 2R$$



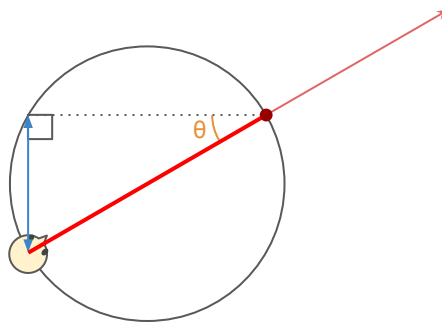
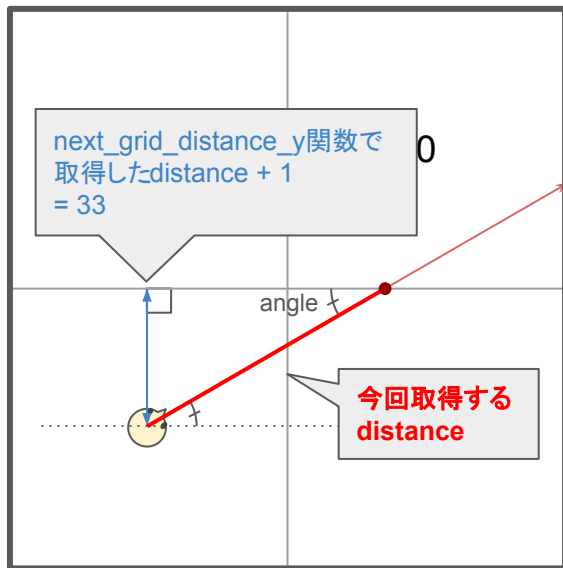
タレスの定理

半円に内接する三角形は直角三角形である。



get_y_step / get_x_step

目的: 次のグリッドまでの距離に角度を考慮して、レイの長さを取得する



円の中心を通る三角形は必ず直角三角形になる

直角三角形の外接円の直径を求める公式は

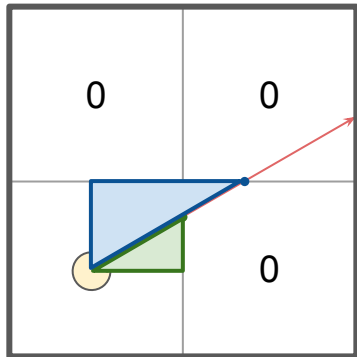
$$\text{直径} = \text{対辺} \div \sin\theta$$

increment_ray_length

目的: 次のグリッドまでの距離をレイの長さに加算する

1. get_y_step と get_x_step で得た距離の内、短い方を採用する
2. レイの向きに応じてレイの長さに加算 / 減算する

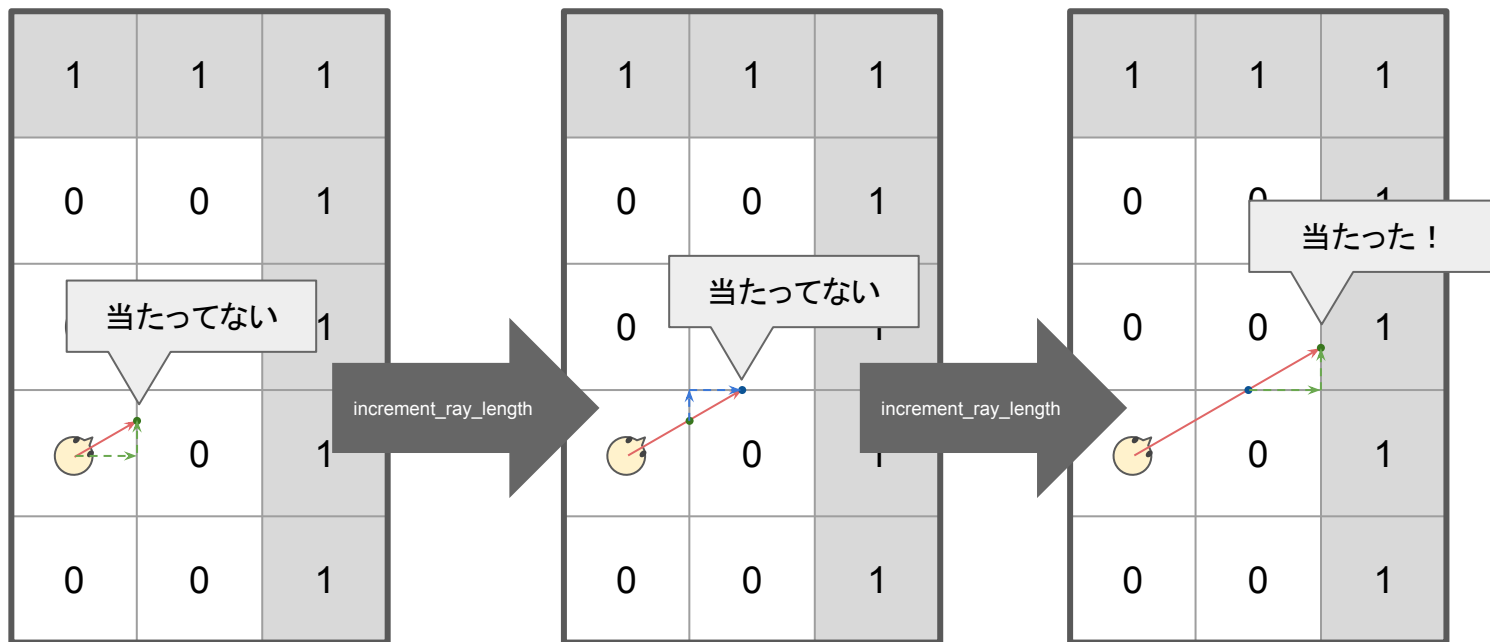
※step_yは壁に対して必ず水平、step_xは壁に対して必ず垂直



step_yとstep_xではstep_xの方が短いのでstep_xに基づいてレイを更新

check_wall

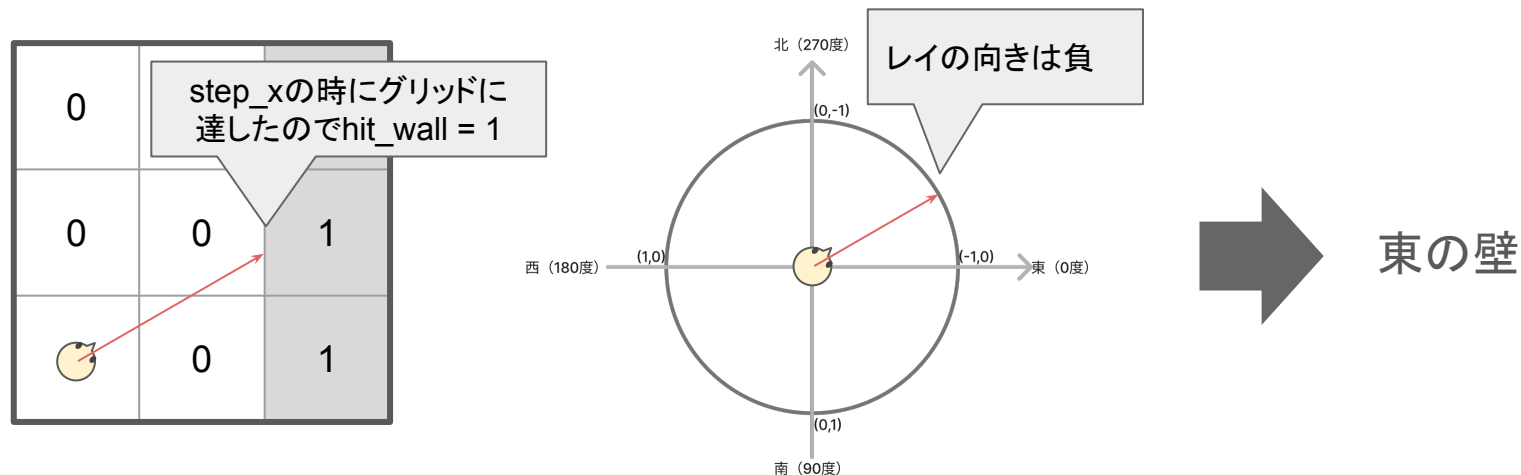
目的: レイが壁に当たっているか確認する



wall_collision

目的: 光線が当たった壁の向きを判定する

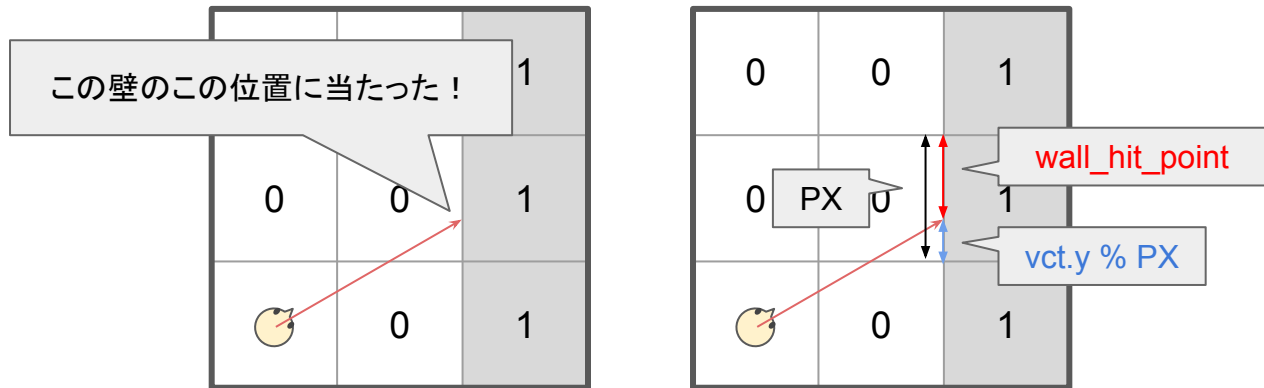
1. ray->hit_wallのフラグにより、垂直方向で当たっているか確認
2. レイの向きが正か負か確認



wall_point

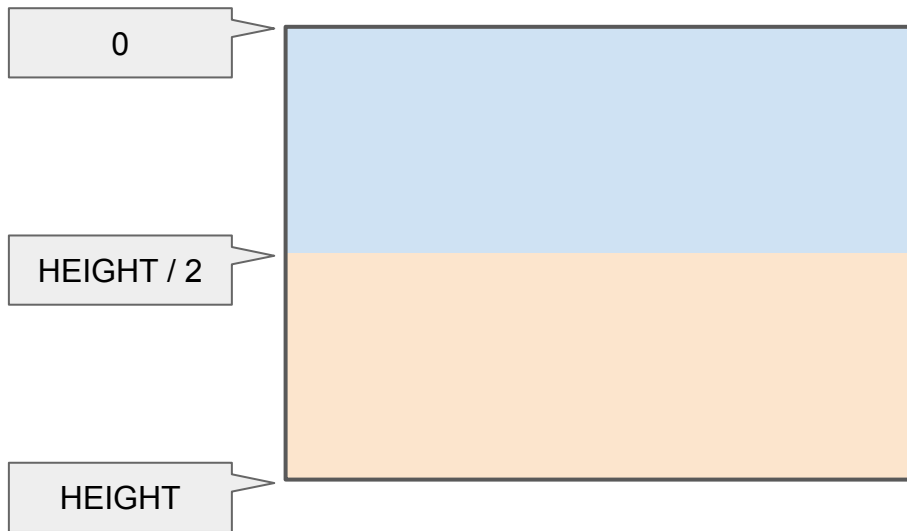
目的: 光線が壁にあたった位置を計算

※後々テクスチャの色を取得するのに使う



draw_floor_and_ceiling

目的: 上半分を天井の色で、下半分を床の色でベタ塗り



rgb_to_int

目的: RGBカラーを一つの数値に変換

[RRRRRRRR][GGGGGGGG][BBBBBBBB]

赤: 上位8ビット(16～23ビット目)

緑: 中位8ビット(8～15ビット目)

青: 下位8ビット(0～7ビット目)

例)(255, 165, 0)

111111111010010100000000

R: 11111111 (255)

G: 10100101 (165)

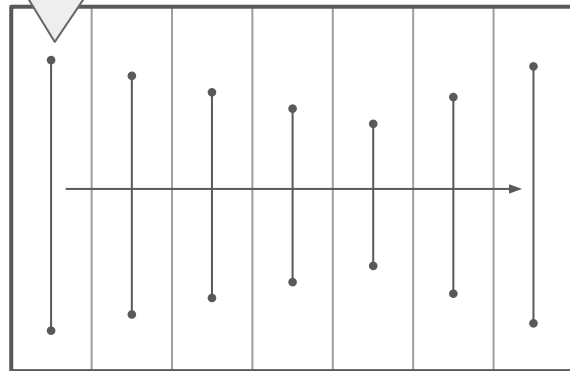
B: 00000000 (0)

draw_walls

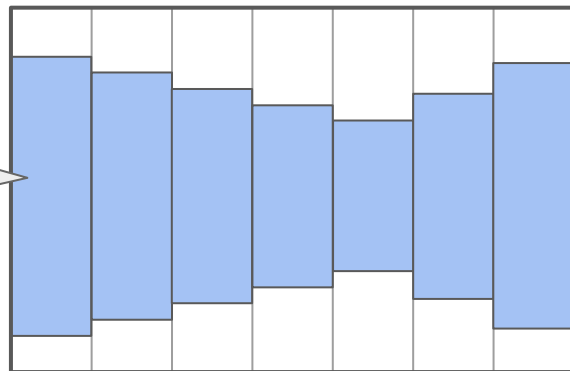
目的: 壁を描画する

1. 左から順に壁の高さを計算する
2. 壁の上下の座標を取得
3. 描画する [draw_wall_column]

1列ずつ壁の高さを確認



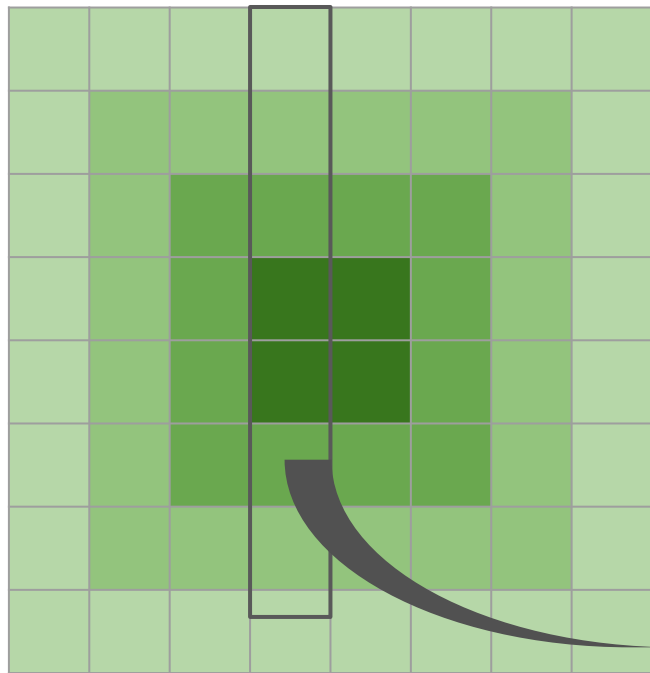
テクスチャから色を取得し壁を描画する



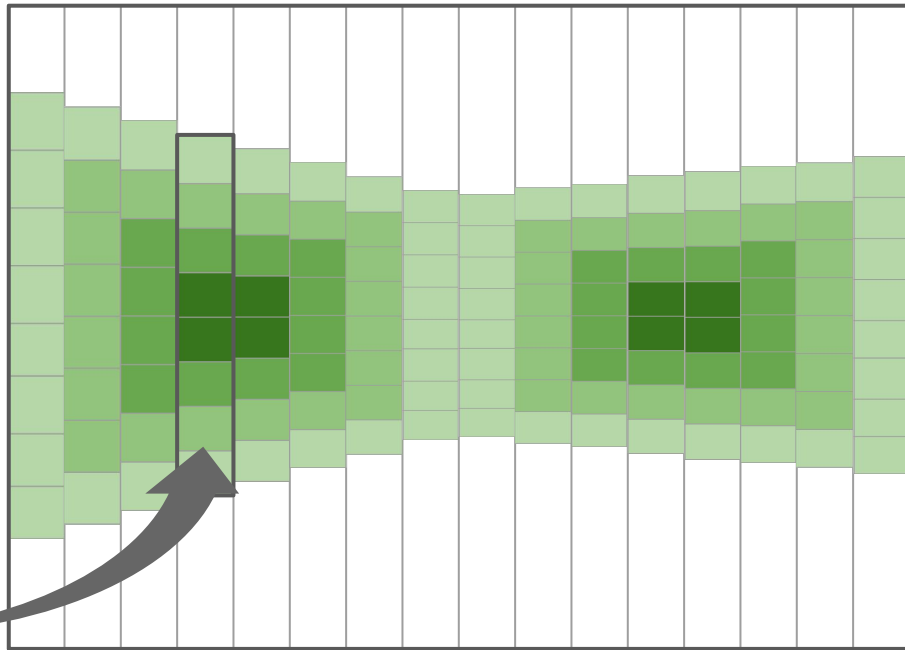
draw_wall_column

目的: テクスチャから色を取得し、描画する

テクスチャ



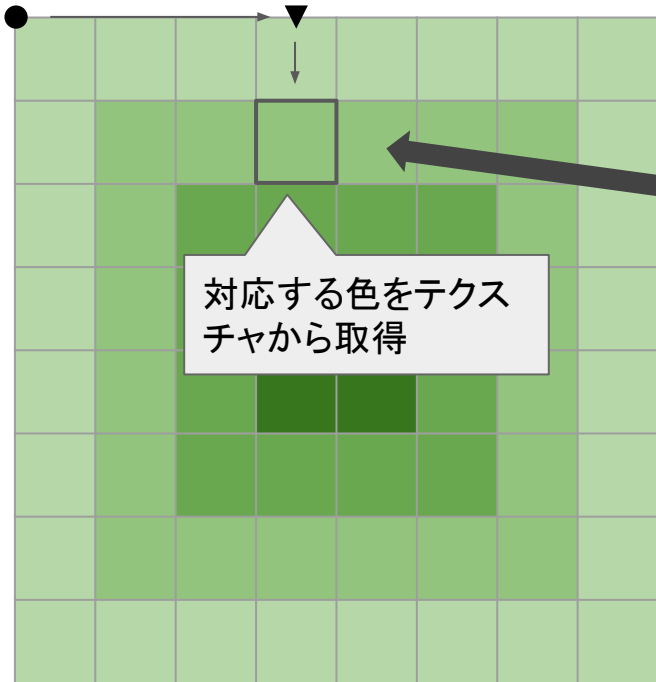
描画イメージ



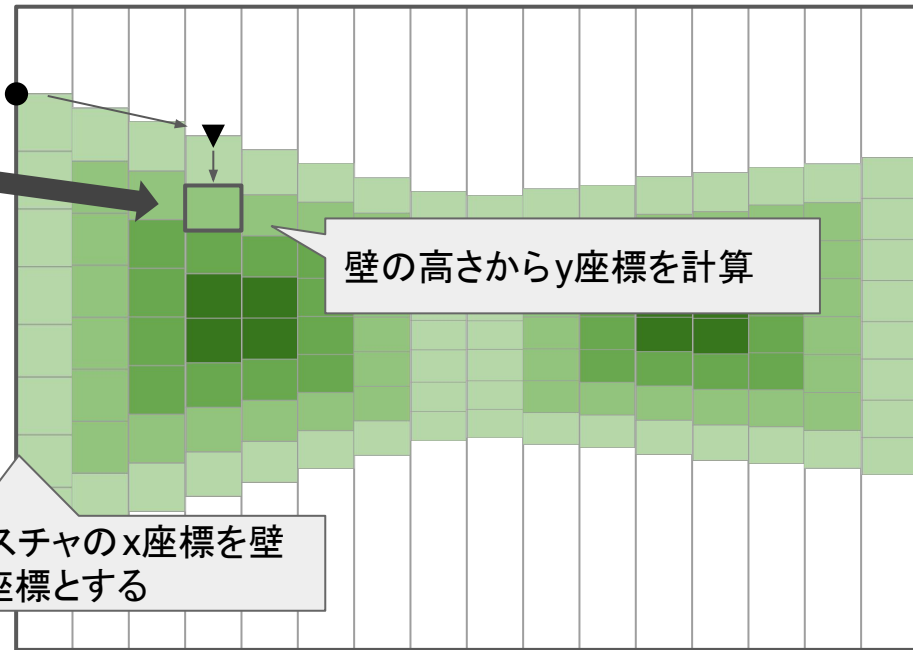
get_texture_color

目的: どのテクスチャを使うか選択し、テクスチャの座標から色を取得する

テクスチャ



描画イメージ



image_pixel_put

目的: 任意のピクセルに色を設定する

1. char *pixelに画像の開始アドレスを設定する
2. 任意のピクセルの位置までオフセットする
3. 色を設定する

img->addr .00000000 00000000 00000000 00000000 ...

ここに色を設定したい

画像データの開始アドレスを指す

char *pixel .00000000 00000000 00000000 00000000 ...

1. アドレスをコピー

char *pixel 00000000 00000000 .00000000 00000000 ...

2. 任意のピクセルまでオフセット

char *pixel 00000000 00000000 .00FF0000 00000000 ...

3. 色を設定

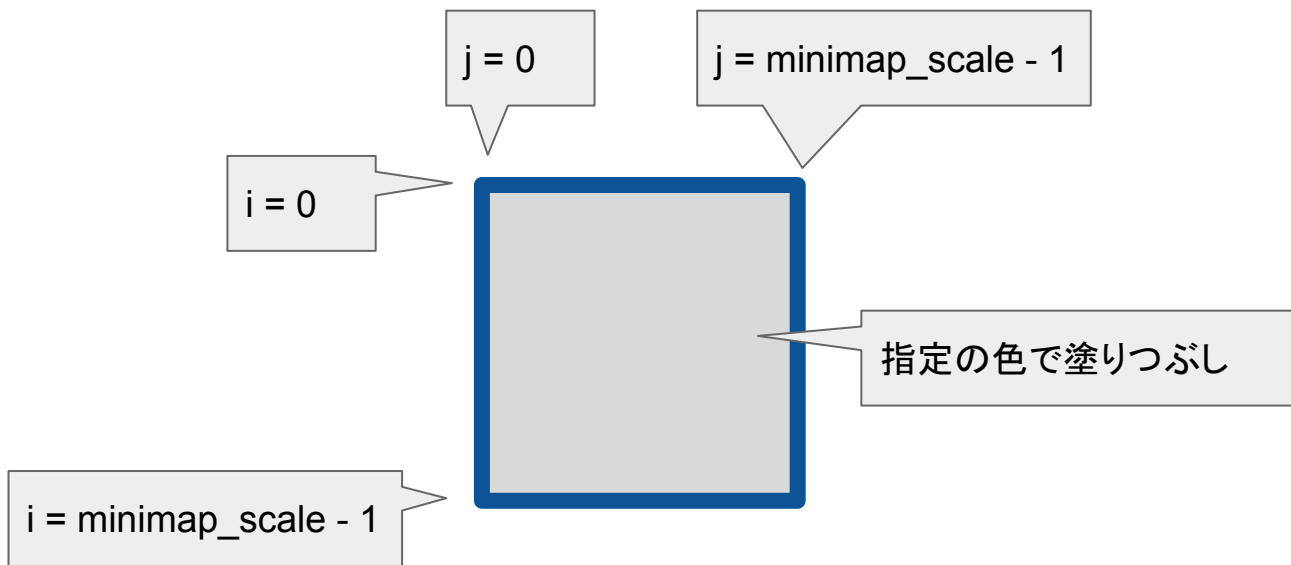
draw_minimap

目的: 2次元のミニマップを描画する

1. ミニマップの縮尺を計算 (マップが大きいほど細くなる)
2. 壁と床をそれぞれ別の色で描画 [draw_grid]
3. プレイヤーを描画 [draw_player]

draw_square

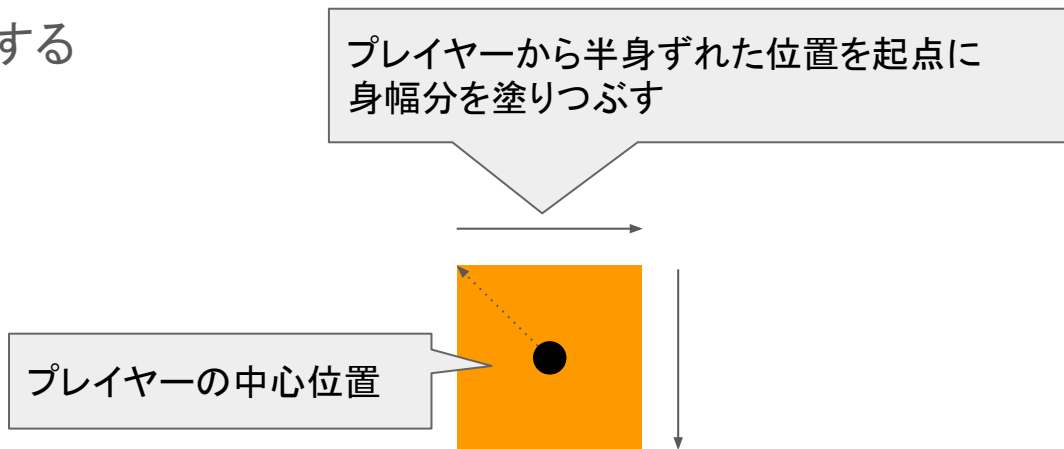
目的: 任意のセルとその周囲を描画



draw_player

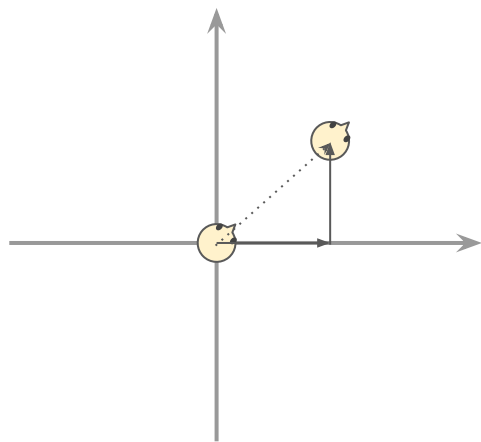
目的:ミニマップにプレイヤー位置を描画する

1. ミニマップ上の座標を取得する
2. プレイヤーを■で描画する



move_player

目的: 移動先の座標が壁でないか確認し、プレイヤーの座標を更新する



移動先のセルが壁ではない

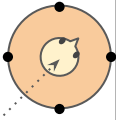

→ 座標を更新

移動先のセルが壁である

→ 移動しない

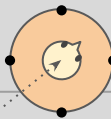

check_hit_wall

目的: プレイヤーの身幅を考慮して壁に当たっているかを確認する

1	1	1
0	0	1
0		1
	0	
0	0	1

プレイヤーの中心座標から前後左右の4点が壁のセルに達していないか確認する

移動OK!

1	1	1
0	0	
0		1
0	0	
0	0	1

移動NG!